DTIC FILE COPY

(4)

WSRL-TN-47/89

AR-005-939

AUSTRALIA

AD-A216 640

A 68000 BASED CARD FOR AN IBM-PC

G.M. O'CONNOR

COMBAT SYSTEMS DIVISION
WEAPONS SYSTEMS RESEARCH LABORATORY

Approved for Public Release.

DSTO AUSTRALIA

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

CONDITIONS OF RELEASE AND DISPOSAL

UNCLASSIFIED

TECHNICAL NOTE
WSRL-TN-47/89

A 68000 BASED CARD FOR AN IBM-PC

G.M. O'Connor

ABSTRACT(U)

This document serves to describe in detail the design and operation of a 68000 based microprocessor card which is connected to an IBM-PC expansion slot.

Author's address:
Combat Systems Division
Weapons Systems Research Laboratory
PO Box 1700, Salisbury
South Australia

Requests to: Chief, Combat Systems Division

UNCLASSIFIED

TABLE OF CONTENTS

1. INTRODUCTION

This document is divided into three main sections. It aims to:

- explain the need for a micro-processor based card to be used within an IBM-PC environment;
- provide a detailed description of the hardware design;
- describe the firmware development, both in technique and content.

2. THE NEED FOR SUCH A CARD

Single quote

As a longer term requirement, it was necessary to provide a simulation of a Tactical Display. Due to the popularity of the IBM-PC clone market and the availability of many software tools tailored towards system development, an IBM clone was chosen as the host computer.

The host computer (IBM-PC) could provide the following environment:

(a) a solid base for hardware development, that is, a well-documented and proven input-output channel (IOC), which could be used to both power and communicate with the card in question; and

(b) an excellent software development platform. Many compilers and cross-compilers are available for IBM-PCs. This allowed software which was to run on the PC and run in a 68000 based environment, to be developed on the PC.

It was envisaged that the IBM-PC would act as a host and thus would not be required to do any major computations. The PC would merely accept tactical data through its serial port and transfer it to the 68000 based card for processing. As such the card was required to be both "XT and AT compatible". This meant that data transfers to the card were to be 8 bit wide.

A Tactical Display requires a range of different displays to be generated. These include icons, menus, and radar signatures. The 68000 based card would primarily provide radar image information and control a graphics processor, however its function could be extended to generate other parts of the Tactical Display if spare CPU time were available. For now, this discussion will be restricted to the 68000 CPU card which has its own buffered bus for interfacing to other peripherals such as graphics processors.

Keywords: Microcircuits, Chips Electronics, 68000 Microcircuits, Cards (Electronics), Australia. (AW)

3. HARDWARE DESIGN

In order to develop a hardware design, it was necessary to consider the tasks required of the CPU card. It was also necessary to choose the micro-processor based on the tools available.

These requirements are:

(a) a fast, simple interface between IBM-PC and the CPU card;

(b) a means of communication with the CPU card which DID NOT involve the PC (for debugging purposes);

(c) system memory for the CPU card to operate in; and

(d) a separate buffered bus interface to allow other peripherals to be controlled by the CPU card (rather than the PC).

It was the microprocessor development tools that determined which CPU would be chosen. A 68000 based Micro-processor Development System (MDS) was available as part of the TEK 8560 Unit. The MDS contains all the features required to develop a 68000 based system. Another feature of a 68000 CPU is that the code is upwardly compatible, and so can become the basis for a 680X0 system with slight software modification.

3.1 The micro-processor development system

The TEK 8560 Unit, supports a 68000 based MDS and was available for use as the development tool. The system acts as a terminal to the TEK 8560 and contains a pod which connects in place of the micro-processor. The MDS provides an editor, assembler, linker, debugger, and input-output routines (eg keyboard scan, screen output and file access). These features allow code to be developed, tested, and placed into Read Only Memory (ROM) so that the system can eventually operate autonomously. The software side of the MDS will be covered in Section 4 of this document, however as the MDS is a hardware, as well as software tool, some of these hardware aspects will be discussed below.

The MDS has three modes of operation.

(a) The first allows the user to generate code on the MDS with no external hardware connected to the pod. In this way, the system acts as a 68000 based host. It has no real use except as a learning tool.

(b) The next mode is by far the most useful for hardware development. The user can generate code but still has all the I/O features inherent in the system. As such, all input and output can be simulated until the system has been fully implemented. The hardware must be present while operating in this mode, and the MDS can be set up so as to partition memory into MDS memory and development system memory. (This feature is extremely useful when the ROM in a system needs to be modified. The MDS memory can be made to superimpose the read only memory while modifications are being carried out.)

(c) The third and final mode is the same as the second except that the MDS I/O routines are not available to the user. This mode is used to ensure the card can run as a stand-alone unit.

3.2 A simple interface

The requirement was for a communications channel between the host IBM-PC and the 68000 based card. A dual port memory configuration was chosen as it provided a high speed buffer interface. The availability of large dual port memory chips in recent years has made it possible to buy 8 K x 8 chip modules. These static devices require no refresh and have two separate address and data paths to the same memory cells. The Integrated Device Technologies (IDT) brand devices chosen have two BUSY signals. These signals are generated by an internal arbitration system so that whenever a contention occurs, due to both systems accessing the same cell, one takes precedence. The BUSY signal is used to delay either system.

The PC BUSY signal was tri-stated and brought active low whenever contention occured. This signal was then fed into IOCHRDY on the PC bus. The IOCHRDY signal is available on the PC bus to allow slow peripherals to

delay the synchronous 80X8X processor by one or more clock cycles. The Chip Enable and Output Enable signals were tied together to minimise decoding on the card.

The 68000 CPU is an asynchronous processor and requires the hardware to provide a Data ACKnowledge signal (DTACK) in order to end its current cycle. This signal will be discussed in greater detail in Section 3.3.

3.3 An independent communication channel

While communications were required between the 68000 CPU and the PC, it was necessary to include some form of independent interface for debugging purposes. This came in the form of a serial terminal, which allowed a user to monitor or control the card, and which could be removed simply, once the CPU card was operating in a dedicated system. A serial interface was designed in a piggyback format which, in conjunction with the appropriate software, enabled communications to take place. This software would reside primarily in the monitor, which will be discussed in Section 4.

The serial interface was adapted from a Force brand design because the Force 68000 card had a simple monitor/debugger in ROM. At that time no other option was available but to write a monitor. This would have been a lengthy procedure. As a result, the Force monitor was used, and the serial interface was adapted so that no software changes would be necessary. The monitor software will be discussed in greater detail in Section 4 of this document.

The piggyback serial card is a simple design using a 6850 Asynchronous Communications Interface (ACIA). The 6850 provides the data formatting and control to interface serial asynchronous data communications information to 8 bit bus systems. The 68000 CPU was however designed to accept 68XX peripherals, and accepts signals from the ACIA. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control.

As mentioned previously, the 68000 generates all the controls necessary to interface with the 68XX family of synchronous peripherals. These devices require an enable (E) signal (usually the phase-2 clock of 68XX systems), which defines the periods of data transfer to and from the processor. They also require read/write (R/W), chip select (CS) and register select (RS) control signals.

For each system application of the 68000, bus cycles are likely to have different lengths. Therefore if a constant frequency clock is used to drive E on the peripherals, there must be a guarantee that data is transferred with respect to the clock. This requirement is not always met in asynchronous bus systems. The 68000 does this : when the peripheral address space is decoded, the VPA signal is asserted, instead of the normal handshaking with DTACK. This signals the processor to become compatible with the 68XX family by waiting for the proper phase of E and then asserting the VMA signal. The address and R/W lines will already be valid. If the sequence begins too late with respect to the phase of E, all address and control signals will remain stable until the next cycle, when compatible transfer will be ensured.

In 68000 systems, the VMA signal used in the chip select equation of all the 68XX family peripherals. During references to a peripheral, it meets all timimg requirements of a chip-select input.

The serial interface uses an erasable programmable logic device, the EP900 manufactured by Altera. Using a schematic capture package, the logic

required to generate all the necessary signals to interface the ACIA to the CPU was generated. The twenty-three address lines were decoded to select one of two addresses. These are two consecutive upper 8 bit addresses which select either of four registers depending on the R/W line.

These registers are assigned as follows.

ADDRESS:	MODE:	DESCRIPTION:
0C0080	Read	Status Register
0C0080	Write	Control Register
0C0082	Read	Receive Data Register
0C0082	Write	Transmit Data Register

The memory addresses are the same as those used by Force, therefore the code was fully compatible.

As already stated, the 6850 requires a synchronous interface with the CPU. The decoded address was ANDed with the VMA signal and the upper data strobe (UDS) to generate the chip select. (Data is valid on the falling edge of UDS for the upper 8 bits of the data bus.)

The synchronous peripherals are also required to return an acknowledge signal in the form of VPA. The VPA was generated by ANDing the decoded address with the address strobe (AS). To allow for multiple 68XX peripherals, the VPA signal was tri-stated.

Inputs and outputs to the RS-232 line are made via standard drivers and receivers. Jumpers were provided to allow for different configurations.

The baud rate generator was designed using a COM8116 chip. The baud rate is derived from a 5.0688 MHz crystal oscillator, and uses four switches to divide the clock down to the required transmission rate. This signal is fed into the ACIA.

3.4 System memory

Static memory was chosen for two reasons:

(a) it did not require any refresh circuitry; and

(b) access times would allow the 68000 to run at maximum speed with no wait states.

A pair of 64 k word (16 bit wide) modules was used, to give a total of 256 k bytes of fast access static RAM. These single-in-line, 40 pin packages are available with access times down to 70 ns, however 100 ns access time devices were chosen. The modules have been designed for use with 680X0 systems as they contain an upper byte and lower byte control line, which can be connected directly to UDS and LDS signals on the CPU.

Also required was read only memory (ROM). This memory was necessary to put the processor into a known state, and also to execute programs without the need that they be loaded into dynamic memory every time power is applied to the circuit. The ROMs chosen were 32 k byte devices. Two memory chips were used: one for the upper eight bits and one for the lower eight bits.

All decoding for the memory was generated using an EPLD from Altera. The device chosen was the EP1800. The address lines were decoded, ANDed with either UDS or LDS, and with ADS. To generate the DTACK signal, similar

decoding to the chip select was produced, and ANDed with the BUSY signal. In this way, a DTACK was produced for all memory chips within the assigned memory space.

The DTACK signal, which determines the completion of a cycle, was generated for RAM and ROM devices. The signal was however delayed for slow access ROMs. This was accomplished by using a shift register which was clocked by the CPU clock, and whose output delay could be controlled via jumpers. The delayed DTACK is tri-stated in order to allow for other additions to the system, and fed to the CPU.

The memory was decoded so that the first eight bytes of ROM, which reside at location 80000 hex, are mapped to 00000 hex. This was done because when the CPU is reset, its program counter and stack pointer are read from these first eight bytes, and program execution is determined from these.

3.5 A separate buffered bus

As the 68000 CPU card was designed to control other peripherals, an independent bus was needed. The bus was designed to be flexible , with all the handshaking signals, address bus, data bus, and power supplies present. The expansion bus uses two header plugs, which are placed on the top side of the card. These 40 and 26 way plugs can then be connected to other peripherals via ribbon cable.

Seven interrupt lines are supplied, and only autovectoring is supported. (A vector is always taken from the predefined vector table within the 68000 memory space whenever an interrupt line is brought low.) The autovectoring sequence requires the VPA signal to be brought low with the advent of an interrupt acknowledge.

When an interrupt line on the bus is brought low, it is encoded via a priority encoder programmed into the Altera EP1800. The three encoded lines are sent to the interrupt inputs IPL0, IPL1, IPL2 on the CPU. (A zero indicates no interrupt request.) Interrupts arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception process is postponed.

The interrupt hardware handshaking is as follows.

(a) The external hardware requests an interrupt by driving one or more of the priority encoded IPL lines low on the EP1800.

(b) The CPU compares interrupt level in the status register and waits for the current instruction to complete, places the interrupt level on address lines A1, A2, A3, and sets all the function code lines FC0-FC2 high to indicate an interrupt acknowledge. The AS line is then asserted.

(c) The EP1800 produces an INTACK signal on ADS and CPU status lines FC0-FC3 asserted, and also produces a low to VPA to signify an autovectored exception.

(d) The processor then commences the interrupt handling routine.

The bus also allows a bidirectional RESET signal to be asserted. If a software reset is performed, the CPU will reset any external devices. The IBM-PC reset is also capable of resetting the CPU and peripherals, as is the case when the reset switch provided on the card is depressed.

4. FIRMWARE DEVELOPMENT

There were three requirements of the firmware:

(a) supply a start address and stack pointer to the CPU at reset;

(b) provide a debugger/monitor for hardware and software checks; and

(c) allow code to be loaded and executed directly via the interface to the IBM-PC.

The MDS was used to develop the software to be put into ROM. The Force ROM boot address was modified so that rather than boot up directly into the monitor, the system would start 4000 hex bytes further on. An EPROM programmer was used to modify the boot address, and the new version of the Force ROMs were burned into larger memory devices (to eventually accommodate the extra code) and placed into the CPU card. The emulator was then set up to super-impose its system RAM over the ROM area (4000 hex bytes on) so that code could be developed in the same memory in which it was to eventually reside.

In the new boot sequence, the card now prompts for one of five choices from the serial interface rather than enter the monitor.

1. Enter FORCE monitor
2. Load source from PC file in MOTOROLA-S format
3. Load source from PC file in BINARY format
4. Save source to file on PC in BINARY format
5. Load and Execute source from PC file in MOTOROLA-S format

Hit any key to select from menu ..

Except for the first choice, the card requires additional software on the PC end to operate correctly.

The monitor start address was known, however it was again necessary to modify the monitor in order to exit it. The emulator was used to trace through monitor command sequences. It was obvious that the code was reading the entered string, then branching off to the address at its offset in a table. A little used instruction was sacrificed and its offset was changed to point to a return address within the new software. The instruction label was also modified to "EX" for EXit. The monitor could then be used as a sub program of the menu.

The code that is associated with the menu includes many subroutines, such as:

- independent screen and keyboard I/O handling (to the Force routines);
- string handling;
- a Motorola-S to binary loader;
- loaders to pass binary data to and from files on the IBM-PC; and
- a Motorola-S "Load and Go" routine.

The code was all developed on the MDS in 68000 assembly code, and then transferred to the PC as one big binary file (including modified Force monitor). The binary transfer utility was actually used as the transfer medium to the PC. An EPROM programmer connected to the IBM-PC was then used to burn the EPROMs. Few iterations were needed to produce the final ROMs, as the MDS could be used to test all the functions while in Mode 3.

Software developed for the CPU card can be developed via the Force monitor, then saved and reloaded from disk on the IBM-PC. It is however more convenient to use a cross compiler or cross assembler based on the PC to generate the software, then down-load it to the card using a Motorola-S record. (Most 68000 compilers produce this form of output.)

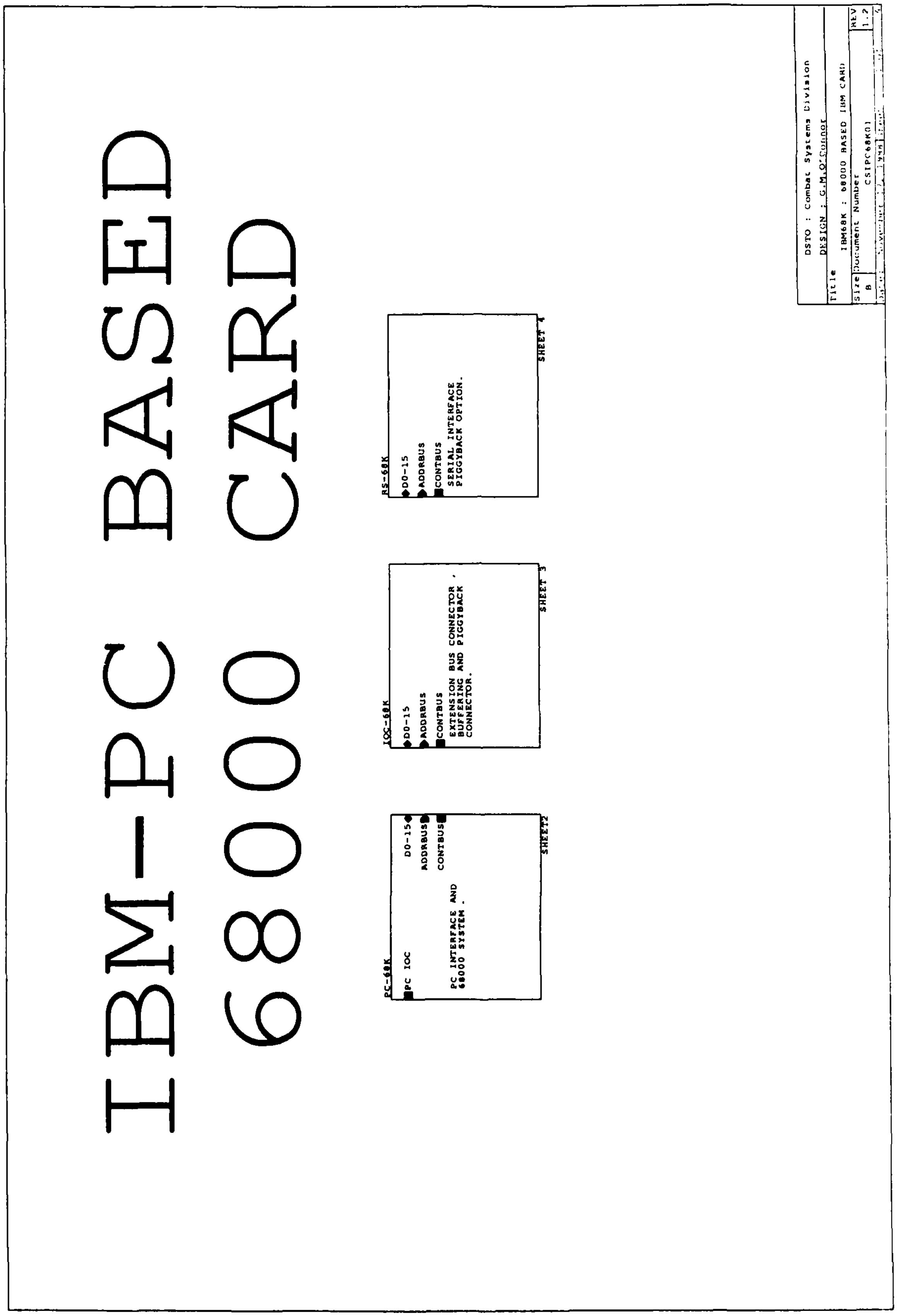
The PC software is an executable file. It prompts the user for an interactive session with the card, or direct "Load and Go".

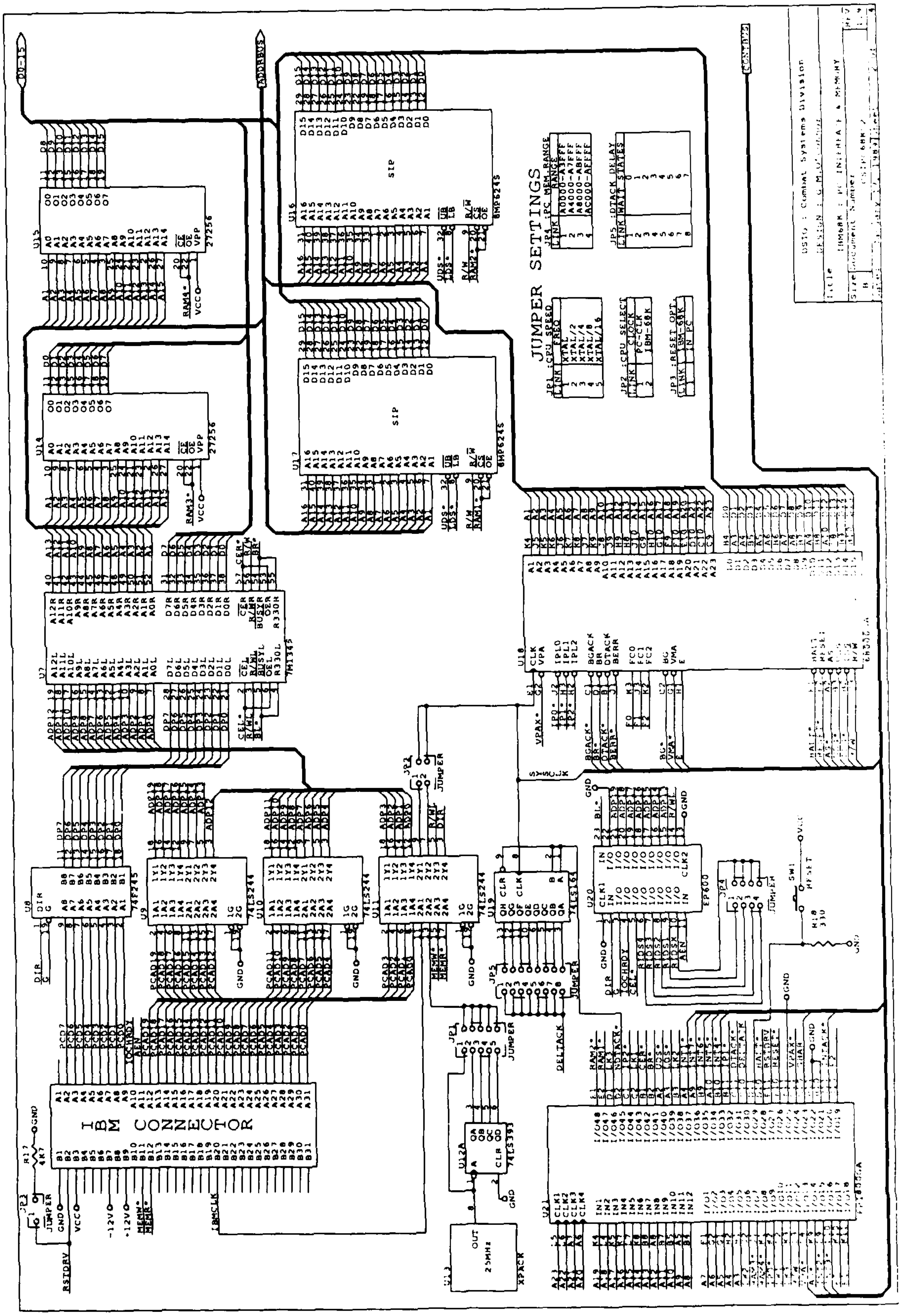
REFERENCES

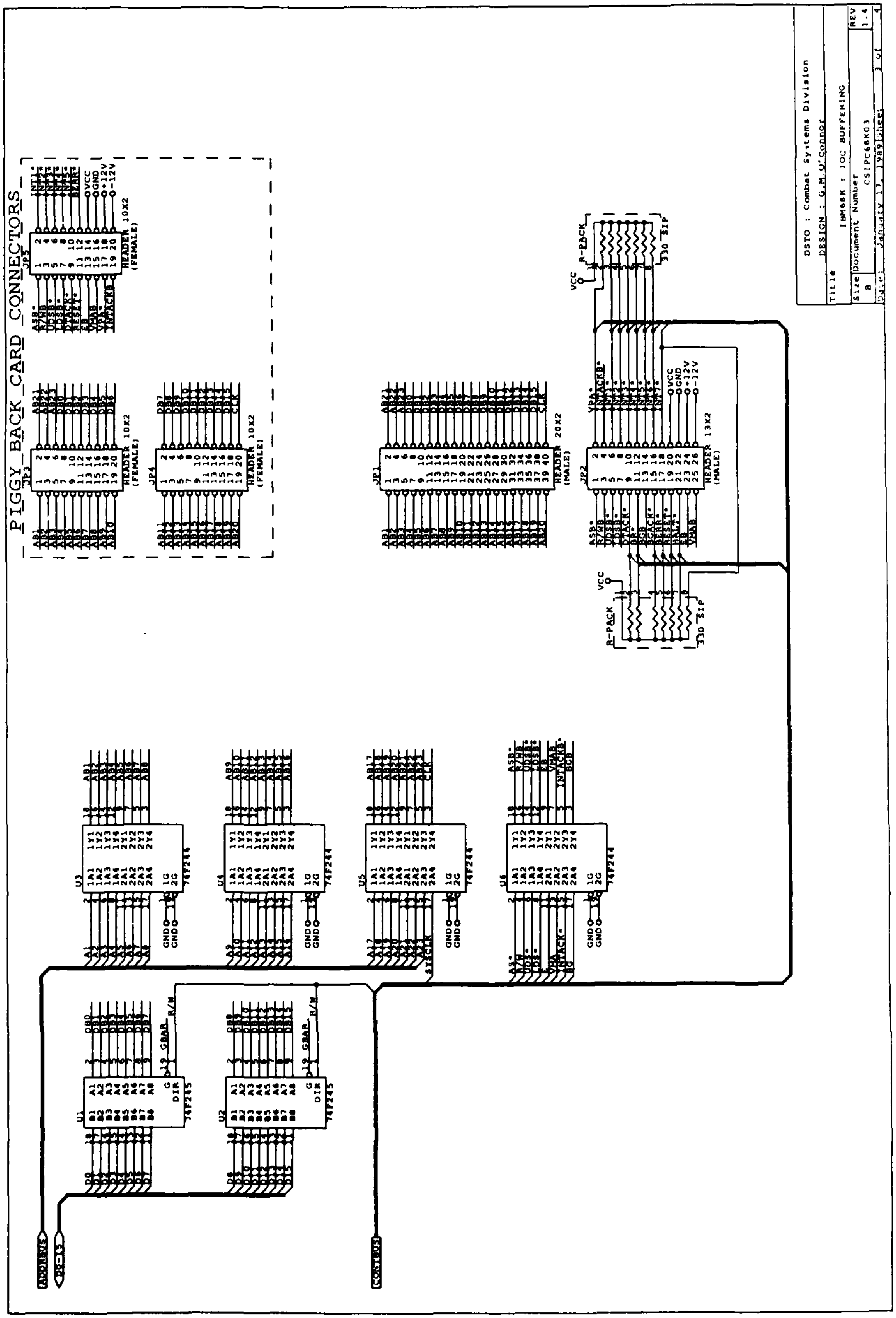
No.	Author	Title
1	-	"MC68000 16/32-Bit Microprocessor Hardware Manual". Motorola Semiconductors, Austin, Texas, March 1985
2	-	"MC68000 16/32-Bit Microprocessor Programmer's Reference Manual". Motorola Semiconductors, Prentice-Hall, 1984
3	-	"MC6850 Asynchronous Communications Interface Adapter Data Sheet". Motorola Semiconductors, Austin, Texas, October 1978

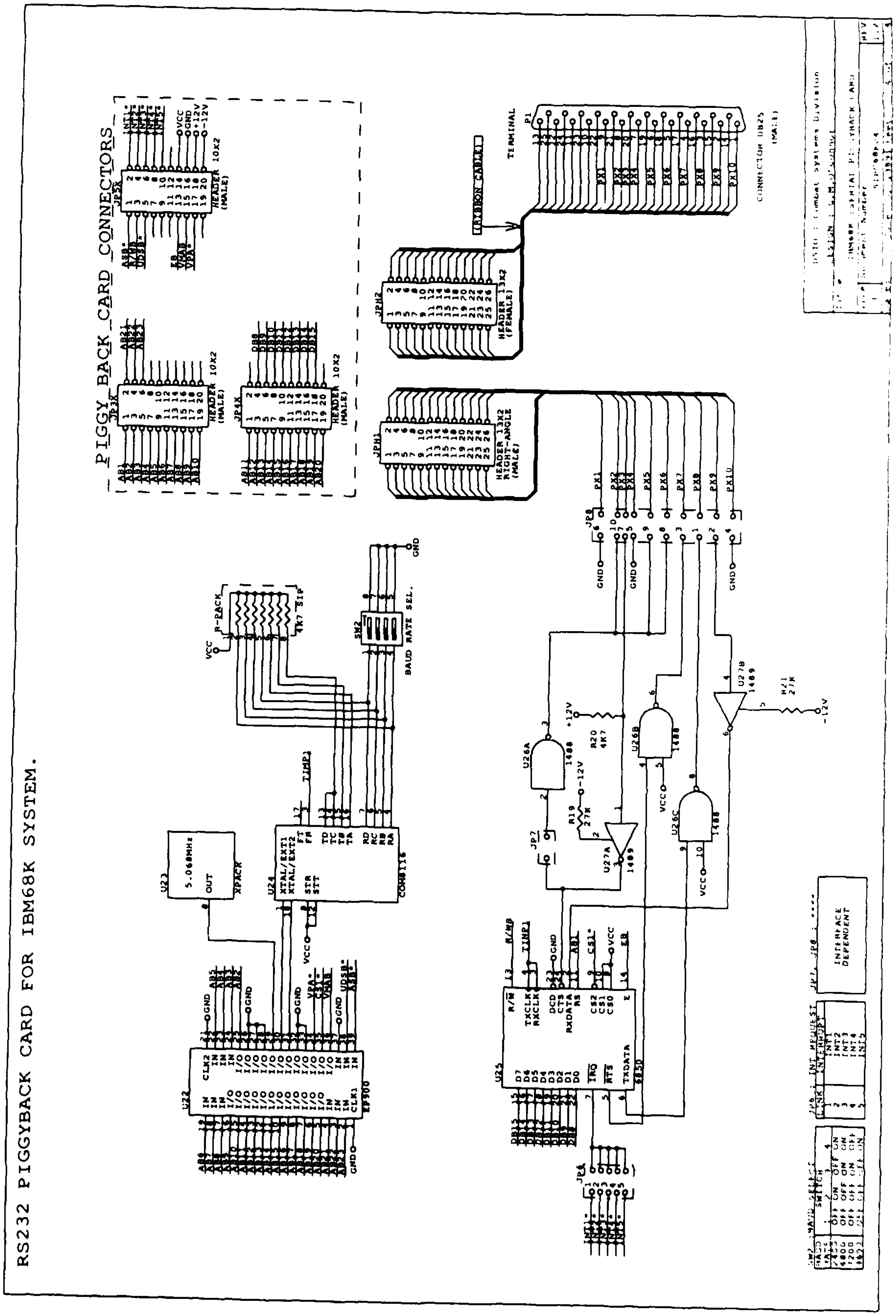
ANNEX A

CIRCUIT DIAGRAMS


IBM-PC BASED
68000 CARD
PC-68K
PC IOC
PC INTERFACE AND
68000 SYSTEM .
D0-15
ADDRBUS
CONTBUS
SHEET2
IOC-68K
D0-15
ADDRBUS
CONTBUS
EXTENSION BUS CONNECTOR ,
BUFFERING AND PIGGYBACK
CONNECTOR.
SHEET 3
RS-68K
D0-15
ADDRBUS
CONTBUS
SERIAL INTERFACE
PIGGYBACK OPTION.
SHEET 4
DSTO : Combat Systems Division
DESIGN : G.M.O'Connor
Title
IBM68K : 68000 BASED IBM CARD
Size
B
Document Number
CSIPC68K01
REV
1.2


JUMPER SETTINGS
IBM CONNECTOR
EP600
74LS244
74F245
27256
8MP624S


PIGGY BACK CARD CONNECTORS
DSTO : Combat Systems Division
DESIGN : G.M.O'Connor
IHM68K : IOC BUFFERING
CSIPC68K03
74F244
74F245
HEADER 20X2 (MALE)
HEADER 13X2 (MALE)
HEADER 10X2 (FEMALE)
330 SIP
R-PACK


RS232 PIGGYBACK CARD FOR IBM68K SYSTEM.
PIGGY BACK CARD CONNECTORS

DISTRIBUTION

	No. of copies
Defence Science and Technology Organisation	
Chief Defence Scientist	1
First Assistant Secretary Science Policy	
First Assistant Secretary Science Corporate Management	
Director General Science and Technology Programs	
Counsellor, Defence Science, London	Cnt Sht Only
Counsellor, Defence Science, Washington	Cnt Sht Only
Defence Science Representative, Bangkok	Cnt Sht Only
Defence Adviser, Defence Research Centre, Kuala Lumpur	Cnt Sht Only
Weapons Systems Research Laboratory	
Director, Weapons Systems Research Laboratory	1
Chief, Combat Systems Division	1
Head, Combat Systems Integration	1
P. Tyler, Combat Systems Integration	1
T.J. Mellor, Combat Systems Integration	1
Author	1
Aeronautical Research Laboratory	
Director, Aeronautical Research Laboratory	1
Electronics Research Laboratory	
Director, Electronics Research Laboratory	1
Materials Research Laboratory	
Director, Materials Research Laboratory	1
Surveillance Research Laboratory	
Director, Surveillance Research Laboratory	1
Libraries and Information Services	
Librarian, Technical Reports Centre, Defence Central Library, Campbell Park	1

Document Exchange Centre Defence Information Services and Science Liaison Branch for:	
Microfiche copying	1
United Kingdom, Defence Research Information Centre	2
United States, Defense Technical Information Center	12
Canada, Director, Scientific Information Services	1
New Zealand, Ministry of Defence	1
National Library of Australia	1
Main Library, Defence Science and Technology Organisation Salisbury	2
Library, Materials Research Laboratory	1
Library, DSD, Melbourne	1
Library, Maritime Systems Division	1
Australian Defence Force Academy Library	1
Department of Defence	
Director of Departmental Publications	1
Joint Intelligence Organisation (DSTI)	1
UNITED KINGDOM	
Institution of Electrical Engineers	1
British Library Document Supply	1
Spares	3
Total number of copies	43

DOCUMENT CONTROL DATA SHEET

Security classification of this page : UNCLASSIFIED

1 DOCUMENT NUMBERS	**2 SECURITY CLASSIFICATION**
AR Number : AR-005-939	a. Complete Document : Unclassified
Series Number : WSRL-TN-47/89	b. Title in Isolation : Unclassified
Other Numbers :	c. Summary in Isolation : Unclassified
	3 DOWNGRADING / DELIMITING INSTRUCTIONS

4 TITLE

A 68000 BASED CARD FOR AN IBM-PC

5 PERSONAL AUTHOR (S)	**6 DOCUMENT DATE**
G.M. O'Connor	July 1989
	7 7.1 TOTAL NUMBER OF PAGES: 13
	7.2 NUMBER OF REFERENCES: 3

8 8.1 CORPORATE AUTHOR (S)	**9 REFERENCE NUMBERS**
Weapons Systems Research Laboratory	a. Task : NAV 87/225
	b. Sponsoring Agency : NAVY
8.2 DOCUMENT SERIES and NUMBER Technical Note 47/89	**10 COST CODE**

11 IMPRINT (Publishing organisation)	**12 COMPUTER PROGRAM (S)** (Title (s) and language (s))
Defence Science and Technology Organisation	

13 RELEASE LIMITATIONS (of the document)

Approved for Public Release.

Security classification of this page : UNCLASSIFIED

Security classification of this page : UNCLASSIFIED

14 ANNOUNCEMENT LIMITATIONS (of the information on these pages)

No limitation

15 DESCRIPTORS

a. EJC Thesaurus Terms

IBM-PC
Microprocessors

b. Non - Thesaurus Terms

68000 CPU card

16 COSATI CODES

0062A

17 SUMMARY OR ABSTRACT
(if this is security classified, the announcement of this report will be similarly classified)

This document serves to describe in detail the design and operation of a 68000 based microprocessor card which is connected to an IBM-PC expansion slot.

Security classification of this page : UNCLASSIFIED